computing sciences

Form: PAS[illegible]-62-0

SBC SYSTEMS REFERENCE MANUAL

IBSYS OPERATING SYSTEM

This reference manual describes the basic information necessary for submitting a job to be processed by the IBSYS Operating System. Besides a description of the set-up sheet and the overall characteristics of deck set-up, there are descriptions of input/output routines and subroutines which have been added by SBC.

Changes and minor additions to the manual since its last publication are identified by a vertical line in the left margin. The major additions to this manual are the descriptions of SBC subroutines:

REREAD and REWRIT
FXEM, FXLMS, and FSLES
FFPTS

THE SERVICE BUREAU CORPORATION
1456 CASTRO ON COURT
PALO ALTO, CALIFORNIA

Jan., 1966

SBC

TABLE OF CONTENTS

I. INTRODUCTION

It has been SBC's policy in the past to maintain each IBM Operating System in its standard form, that is, to maintain the system so that it can be used in the manner specified in the associated IBM manuals. In certain instances SBC makes modifications to orient the Operating System to the hardware configuration of it's computer system, or adds features in the form of options which makes the Operating System more efficient, more usable. Occasionally, changes are made which supersede the way the system was originally distributed. Changes of this type are very minor and are made to more completely adapt the system to SBC's "Open Shop" method of servicing our customers.

These policies of maintaining a standard Operating System have not changed for IBSYS. The information contained in this manual is generally an expansion of that mentioned in related IBM manuals and pertains directly to the use of the IBSYS Operating System at the Bay Area Scientific Computer Center. In cases where the information conflicts with what is stated in IBM manuals, explicit words to that effect will be included in the text.

II. IBSYS DECK SET-Up

Of all the IBSYS Monitor control cards that could appear in a job deck, only four play a key role in directing the processing of a job. They are $JOB, $EXECUTE, $IBSYS and $ID. The first two must be present in each job deck. The last two may or may not be present depending on the nature of the job.

To understand the process of setting up a deck for the IBSYS Operating System, it is helpful to understand the difference between a "Job" and a "Job segment".

A Job segment consists of those cards to be processed by a particular subsystem. It can be one of two types; a subsystem segment which starts with a $EXECUTE control card or, an IBSYS Monitor segment which starts with a $IBSYS control card.

A job begins with a $JOB control card and is followed by as many "Job Segment" decks as are necessary to complete the problem.

The $JOB Control Card

This is the first card of each job. When encountered, the input/output unit assignments are automatically restored if needed, output on the system output and system peripheral punch units is separated and identified, the IBM 716 printer clock value is produced on-line, and an accounting pause is produced.

The contents of the $JOB card, punched in columns 16-72, should include:

1) Company name
2) Estimate Number
3) Requestor's name or initials
4) Project Number, if required

Difference between the control cards $JOB and $ID

The $JOB control card is used for interjob accounting purposes and must be the first card of a job.

The $ID control card is optional and would be used for intrajob accounting purposes. If used, the $ID card would be placed in front of each 'Segment deck' and when encountered, causes the output on the system output and system peripheral punch units to be separated and identified and the IBM 716 printer clock value to be produced on-line. Processing then continues.

The contents of the $ID card, punched in columns 7-72, may be anything the user desires.

Use of END-OF-FILE Cards

An End-of-File card is identified by having only a 7/8 multi-punch in column one. The remainder of the card is ignored. Some subsystems require that an end-of-file be the last card of the 'Segment deck', others don't. In order to establish a consistent pattern for setting up decks, SBC has made the necessary modifications so that THE LAST CARD OF EVERY 'SEGMENT DECK' CAN BE AN END-OF-FILE CARD.

Composite Deck Set-Up

For purposes of illustration let's consider a job that contains four segments. The deck would be set up as follows:

```
      $JOB                    identifying information

**
      $EXECUTE     IBJOB
          (IBJOB Control cards, program decks and data)
      END-OF-FILE card

 *
      $EXECUTE     SORT
          (Sort Control Cards)
      END-OF-FILE card

 *
      $IBSYS
          (IBSYS Control cards)
      END-OF-FILE card

 *
      $EXECUTE     IBJOB
          (IBJOB control cards, program decks and data)
      END-OF-FILE card
```

Note that the single (*) indicates the point where the $ID control card could be inserted. The double (**) also indicates the insertion point for a $ID card although it is not necessary at that point for the $JOB card will provide the necessary accounting activities for the first segment.

IBSYS Monitor Control Cards - $DATE and $STOP

These two control cards are mentioned because of their importance in the processing of a stack of IBSYS jobs. The first card, $DATE, supplies the current date for use by all subsystems and is effective for an entire stack of jobs. The second card, $STOP, identifies the end of a stack of jobs and causes IBSYS to terminate processing. Users should refrain from including either of these control cards in job decks. They will automatically be supplied for each IBSYS stack by the Operations department.

III. SET UP SHEET FOR IBSYS JOBS

A Green "IBSYS" Setup Sheet has been developed for all IBSYS jobs. The required information is located in the three top lines and in the box labeled "Customer Specify Disposition of Output". Other types of Information may be as follows:

Specifying Tape Units

The IBSYS Operating System always has eight tapes associated with it; A1 - A4 and B1 - B4. Only those units which fall outside this range need be specified.

No explicit information is required regarding the system output or system peripheral punch units unless for example, more than one part paper is needed for print output.

Unexpected Program Stops and Loops

The "Check if Dump Required" box has two entries for requesting a manual core dump. "At Error Stop" is for non-programmed stops or loops and "At Calc Limit" is for jobs which exceed maximum running time. If the appropriate box is not checked and the condition occurs, the job will simply be removed from the machine.

Notice that some subsystems have the capabilities of producing an automatic core dump when certain types of errors are encountered. At present, these capabilities are fixed and cannot be activated or deactivated from an external source.

IV. IBSYS INPUT/OUTPUT UNITS

An integral part of the IBSYS Monitor is a scheme of associating physical input/output units to specific system functions. Since each subsystem requires a minimum number of input/output devices in order to properly operate, this scheme assures that each subsystem will have the necessary devices available. This scheme also provides for the necessary coordination between subsystems and allows the exact status of all units to be known and maintained at all times.

In addition to the assignment of physical input/output units to specific functions, the IBSYS Monitor also maintains a pool of available input/output units; those which are in excess of the number required to fulfill the specified system functions. The exact number of available units is governed by the hardware configuration of the computer system.

Following is a list of the specific system functions and their physical input/output devices, and the pool of available tapes as provided by the IBSYS Operating System at the Bay Area Scientific Computer Center.

System Unit Functions

Symbolic Name	Physical Unit and Density	Use
SYSLB1	A1 - High	IBSYS Operating System Tape
SYSLB2	No unit assigned	Alternate System Library Tape
SYSLB3	No unit assigned	Alternate System Library Tape
SYSLB4	No unit assigned	Alternate System Library Tape
SYSCRD	Card Reader-Channel A	On-line card reader
SYSPRT	Printer-Channel A	On-line printer
SYSPCH	Punch-Channel A	On-line card punch
SYSOU1	B1 - High	System print output tape
SYSOU2	B1 - High	Alternate System print output tape
SYSIN1	A2 - High	System Input Tape
SYSIN2	A2 - High	Alternate System Input Tape
SYSPP1	B2 - High	System Punch Output Tape
SYSPP2	B4 - High	Alternate System Punch Output Tape
SYSCK1	No unit assigned	System Checkpoint Tape
*SYSCK2	B7 - High	Alternate System Checkpoint Tape
SYSUT1	A3 - High	System Scratch Tape
SYSUT2	B3 - High	System Scratch Tape
SYSUT3	A4 - High	System Scratch Tape
SYSUT4	B4 - High	System Scratch Tape
SYSUT5	No unit assigned	Extra System Utility Tape
SYSUT6	No unit assigned	Extra System Utility Tape

Symbolic Name	Physical Unit and Density	Use
SYSUT7	No unit assigned	Extra System Utility Tape
SYSUT8	No unit assigned	Extra System Utility Tape
SYSUT9	No unit assigned	Extra System Utility Tape

* This unit was assigned as A5 in the distributed version. Note that it is used by Sort for checkpoints and unreadable records and by IBJOB when the Debug Package is being used.

Unit Availability Table

Channel A	Channel B
A5	B5
A6	B6

Subsystem Use of Input/Output Units

Each subsystem and program operating under the subsystem draws the necessary input/output devices from the above two sources. The methods by which this is done can be found in the appropriate IBM manuals and will not be discussed in this Bulletin except in cases where a clarification of available information is needed or changes have been made which supersede the IBM Manual Information.

V. IBJOB PROCESSOR

The IBJOB Processor, itself a subsystem under IBSYS, contains a number of its own individual subsystems. Specifically these IBJOB subsystems are: FORTRAN IV, COBOL, IBMAP, IBLDR, and a library of subroutines, IBLIB.

Changes have been made to the way FORTRAN IV programs use input/output units and, both changes and additions have been made to the library which will primarily benefit FORTRAN IV users.

FORTRAN IV Input/Output Units

FORTRAN IV programs refer to input/output devices by a logical unit number. The source language statement "READ (5, 60) B" illustrates this, where 5 is the number of a logical unit.

The library, IBLIB, contains a File routine for each input/output device on the computer that can be referred to by a logical unit number. These File routines link the logical unit number with a system function or an available input/output unit. In addition, each File routine contains information that describes the way the unit is going to be used; input, output, or both, the mode of the data (BCD or Binary), the blocksize, etc. The following list shows which system unit will be used for each logical unit number and how the system unit is defined as to mode, etc. Refer to the "IBSYS INPUT/OUTPUT UNITS" section for the definitions of the physical units which are assigned to the IBSYS System Units.

FORTRAN Logical Unit	IBSYS System Unit	Mode	Block Size	Use
1	SYSUT1	Binary	256	Input or Output
2	SYSUT2	Binary	256	Input or Output
3	SYSUT3	Binary	256	Input or Output
4	SYSUT4	Binary	256	Input or Output
5	SYSIN1	BCD	14	Input
6	SYSOU1	BCD	110	Output
* 7	SYSPP1	BCD	14	Output
* 8	B(1)	BCD	22	Input or Output
9	A(1)	BCD	22	Input or Output
10	B(2)	Binary	256	Input or Output
11	A(2)	Binary	256	Input or Output

*The file description and/or System unit differs from the distributed version. The last three File routines have been added because of the additional number of input/output devices available to our computer system.

In the list above, A(n) and B(n) refers to the nth unit in the Unit Availability Table for the specified (A or B) channel.

For those FORTRAN IV programmers who do not use the alternate input/output package (ALTIO), a word of caution should be mentioned concerning input/output operations. Generally, logical unit N must be used as it is defined by the File routine. For example, suppose a program uses logical unit 9 as a binary scratch unit; information is written on this unit in binary and then read back at a later time. An error is likely to result if the File routine that is defined above is used, for it specifies logical unit 9 as BCD.

To override any File routine on the library, the user must include the corresponding File routine in his deck, modified to define the input/output device as the program intends to use it. The general form of a File routine would be:

```
Col 1          8          16
    $IBMAP     UNXX       NOLIST, NOREF
               ENTRY      .UNXX.
    .UNXX.     PZE        UNITXX
    UNITXX     FILE       File specifications
               END
```

In the above, XX is the FORTRAN IV logical unit number (with a leading zero if applicable) that is being redefined. Details on how to use the IBMAP pseudo-op FILE can be found in the manual "IBM 7090/7094 Programming Systems: MAP (Macro Assembly Program) Language," form C28-6311.

An example of the File routine for the above program that uses logical unit 9 as Binary would be:

```
Col 1          8          16
    $IBMAP     UN09       NOLIST, NOREF
               ENTRY      .UN09.
    .UN09.     PZE        UNIT09
    UNIT09     FILE       ,A(1), READY, INOUT, BLK=256,
               ETC        BIN, NOLIST
               END
```

General IBLIB Changes

The IBSYS Operating System can operate on either an IBM 7090 or an IBM 7094. Because of this, the distributed Double Precision and Complex mathematic routines simulate these two-word operations as the IBM 7090 doesn't contain the necessary hardware instructions.

To take advantage of the double-word hardware capabilities, all Double Precision and Complex mathematic routines have been reassembled for our IBM 7094. This results in decreasing both execution time and the size of each routine.

Programming Note

Programs containing double-precision or complex computation must be compiled for the IBM 7094. This is accomplished by specifying the 'M94' option on the $IBFTC or $IBMAP card.

SBC Data Plotting Subroutines

The SBC Data Plotting subroutines have been added to the library. A complete description of what they do and how they are used is found in "SBC Systems Reference Manual: 7094 Data plotting Subroutines", Form PASRM-04.

Subroutine CLOCK

This subroutine allows programs to obtain the current value of the IBM 716 printer clock. The calling sequence for a FORTRAN IV program would be

```
CALL CLOCK (X)
```

Upon return from this subroutine, X will be a floating point number that represents the current clock value. The magnitude of the clock value can be 9999.99 where the decimal portion represents hundredths of a minute and the integer represents minutes.

Subroutines REREAD and REWRIT

These routines provide a flexible means of re- formatting data without having to use an external input/output device. This is accomplished by means of a CALL to one of these routines followed by a pseudo-input/output statement.

Using REWRIT

The general form of the two FORTRAN IV statements are:

```
CALL REWRIT
WRITE (N, X) List
```

The information specified in the 'list' is extracted from core, formatted according to FORMAT statement number 'X', and placed in an intermediate BCD buffer. The only value this affords is that its possible to "REREAD" this information using entirely different FORMAT specifications.

Using REREAD

The general form for REREADing is:

```
CALL REREAD
READ (N, X) List
```

This is the more usable of the two routines since re-formatting can be accomplished using data that was read from tape, or written on tape, as well as with data that was REWRITen.

With this routine, the data that was last placed in the intermediate BCD buffer (either by a true BCD READ, a true BCD WRITE, or by a REWRIT) is formatted according to FORMAT statement number 'X' and placed into the variable locations specified by the 'list'.

Rules for using REREAD and REWRIT

In the general form examples above, 'N' represents a logical unit number and can be either an integer or an integer variable. Although this logical unit number is superfluous (because I/O devices are not used) it must nevertheless be a legal number.

The pseudo-input/output statement must immediately follow the CALL and must directionally agree with the routine being used (i.e., READ follows REREAD, WRITE follows REWRIT). Program execution will be terminated with the message "I/O STATEMENT MUST FOLLOW A CALL TO REREAD OR REWRIT" if errors in usage are detected.

Restrictions

Although it's syntactically possible to create a pseudo-input/output statement and associated FORMAT statement to deal with multiple records, neither of these routines can process these types of statements. In other words, for each CALL to either of these routines, only 132 characters of information can be processed.

Subroutines FXEM, FXEMS, FSLES

SBC Subroutines, FXEMS and FSLES are related to the Library Subroutine FXEM which has execution termination control whenever errors are detected by other Library Subroutines. It is essential that the user fully understands the contents of the section titled 'FORTRAN IV UTILITY LIBRARY' in the IBJOB Processor Manual to realize the usefulness of the modifications and additions incorporated. FXEM receives control each time an error is detected by a Library Subroutine. Having received control, it will normally produce an appropriate error message and an error- flow trace on the system output tape.

Before exiting, FXEM determines whether to continue or terminate execution by testing the appropriate bit in control words that are within FXEM. If the tested bit is off, FXEM will terminate execution. If the bit is on, the optional exit message is produced and control is returned to the calling routine which in turn resumes execution. Of all the library subroutines which participate in this scheme, not all provide for resuming execution. Some errors are severe enough that it is considered infeasible to continue.

With the original version of FXEM it was possible for it to produce an infinite number of error messages and traces, provided that the optional exit bits allowed continuation. To protect against a possible loop of this type, SBC has modified FXEM so that it terminates execution after receiving control 100 times. Since this limit may not be totally satisfactory, facilities are provided to change this value during execution.

Besides providing the facilities for changing the count of the number of times FXEM is to get control, SBC's subroutine FXEMS further allows the user to have control over whether or not the error messages, the optional exit messages, and/or the error-flow trace messages are to be produced. Through arguments in the calling sequence each class of message can either be allowed or suppressed. In the case of an error which results in execution being terminated, both classes of messages will unconditionally be produced.

The calling sequence for FXEMS is:

CALL FXEMS (I, J, K)

where I = Integer representing the total number of times FXEM is allowed to receive control before it unconditionally terminates execution. Note that each call to FXEMS, where I is non-zero, will result in redefining the count to the value of I. If the value of I is zero, the current count is unaltered. This allows for the changing of J or K without having to redefine the count.

J = 0	Allow error-flow trace messages.
= 1	Suppress error-flow trace messages.
K = 0	Allow error and optional exit messages.
= 1	Suppress error and optional exit messages.

The IBJOB Processor Manual describes how to change the bits in the FXEM control words which govern the optional exits. In order to provide a convenient way of changing these bits, and to also provide the ability to dynamically change them during program execution, SBC has added subroutine FSLES to the IBJOB library.

The calling sequence for this routine is

CALL FSLES (ICODE1, ICODE2, ... , ICODEn)

where ICODEn = The error code number associated with the error, as defined in the IBJOB Processor Manual under the section "Subroutine Library Error Messages".

If ICODEn is positive, the corresponding control bit will be turned on (take optional exit if possible).

If ICODEn is negative, the corresponding control bit will be turned off (terminate execution).

The argument list is variable, therefore only the pertinent codes need be specified. Optional exit control bits for subroutines with no optional exits can be altered, but they are ignored and execution is always terminated. For those errors which have optional exits, the control bits corresponding to the following error codes are OFF in the distributed system. All other control bits for which there are optional exits are ON.

31, 32, 33, 35, 38, 39, 40, 41, 42, 48, 51, 53, 57, and 65

Subroutine FFPTS

During execution of a FORTRAN IV program in which a floating point underflow or overflow occurs, a floating point trap will also occur. This trap is essentially a transfer to a system routine called .FPTRP. The .FPTRP routine determines the type of underflow or overflow (AC, MQ, or both) and processes it accordingly. A by-product of this processing is a defining message such as 'UNDERFLOW AT XXXXX IN AC'.

Although there is no limit to the number of times this routine can be used, there is a limit to the number of times the messages will be produced (normally 5 messages). When this limit is reached, subsequent processing by .FPTRP will not produce any messages. Since this built-in limit may not be entirely satisfactory, SBC has developed a routine which allows execution time definition of a more desirable limit.

The calling sequence for this routine is:

```
CALL FFPTS (MLIMIT)
```

where MLIMIT = An integer representing the maximum number of messages to be produced. Note that this integer can be zero, in which case no messages will be produced. Note also that each time this routine is called the count is redefined to the value of MLIMIT.

VI. IBM 7090/7094 GENERALIZED SORTING SYSTEM

This IBSYS subsystem is being mentioned not because of any changes that have been made, but rather to clarify its use regarding the order of the merge it can perform.

The order of the merge is limited by the number of tapes available on the computer system. The IBM manual "IBM 7090/7094 Generalized Sorting System: 7090/7094 Sort, "Form C28-6365, states that the number of tapes required by Sort is 2M, where M is the order of the merge.

Under the concepts of monitored operations, certain conventions have to be established regarding the use of certain units (i.e., input, output, Operating System residence). With this in mind, we see that four units are used for these purposes (SYSLB1, SYSIN1, SYSOU1 and SYSPP1) and are therefore not available for general use.

In order to determine what the maximum value of M could be, one must consider the total number of units available on the computer less the four mentioned above. In addition, one must consider whether unreadable records and dictionaries are to be saved and checkpoints are to be taken or not. These Sort features require an additional unit (SYSCK2) if they are desired.

The number of input/output devices at the Bay Area Scientific Computer Center is such that the maximum value of M can be 4 if the NOCKPT option is used, or 3 if checkpoints are to be taken.

SDD - SAN JOSE LABORATORY
Engineering & Scientific
Computation Laboratory - Dept. 555

July 14, 1965

MEMORANDUM TO: All Computer Laboratory Users

SUBJECT: IMPLEMENTATION OF 7090/94 IBSYS OPERATING SYSTEM VERSION 13

PROJECTED DATE OF RELEASE: July 26, 1965

A new version of the IBSYS OPERATING SYSTEM will be implemented. This new version represents many significant improvements to previous versions, the most significant being the introduction of a new FORTRAN IV compiler. The new compiler features speed, language and list output improvements.

Other changes, additions and improvements have been made to the IBSYS Loader, and IBJOB Library, all of which will be discussed individually.

IBSYS Monitor

1. The $SWITCH card no longer changes the densities of the physical units involved.

2. Five additional utility functions (SYSUT5 - SYSUT9) have been added to the SYSUNI function table. These additional utility functions may not be used for IBJOB overlay link-residence.

IBJOB Monitor

1. A new control card ($TITLE) has been added for page headings.

2. Four IBJOB control cards have changed.
   a. The $IBJOB card has an additional parameter 'ALTIO'
   b. The $IBMAP card allows for new IBMAP options. NOSYM, MONSYM, or JOBSYM.
   c. The $IBFTC card no longer contains the 'REF' option. If REF is specified, it will be ignored.
   d. The IBDBL card has new option, 'NOMES'

IBLDR (IBJOB Loader)

1. If 'ALTIO' is specified on the $IBJOB card, an Alternate FORTRAN IV I/O package replaces the standard FORTRAN IV I/O package. This allows 1900 more locations to be available to the user, but suppresses optimal I/O buffering.

2. A new expanded load map facility has been added to the loader, which will eliminate in most cases, the need for a LOGIC printout.

IBMAP (MAP Assembler)

1. Two new pseudo operations are available (LITORG and LOC).

2. A new option on the $IBMAP card to allow pre-definition of IBSYS and IBJOB monitor symbols.

IBFTC (FORTRAN IV Compiler)

1. The compiling time for most jobs has been decreased. The amount of performance improvement varies upon the options requested, the nature of the source program, and the number of compilations within a job.

2. FORTRAN IV now allows (a) up to seven (7) dimensions, (b) non-standard returns from subroutines, and (c) multiple entry points to a subprogram.

3. The PREST option ($OEDIT card) for FORTRAN IV is now ignored.

4. A name of any deck may not be the same as any program, subprogram or ENTRY name in the same job.
The compiler will issue an error message for the case where the deck name is the same as that of the (sub)program name for, or ENTRY name in that deck.

5. Consult FORTRAN IV Manual for VERSION 13 for changes relating to non-executable statement ordering. (i.e., COMMON, EQUIVALENCE, DIMENSION, etc.)

IBCBC (COBOL Compiler)

1. APPLY CHECK-SUM, APPLY SEQUENCE-CHECK, RERUN...ALL FILES, RERUN...OUTPUT FILES, RERUN...INPUT FILES, RERUN options have been implemented.

2. Access to the FORTRAN IV mathematics library is provided through additions to the language.

3. The DISPLAY verb allows greater versatility.

4. Use of the ELSE (OTHERWISE) option following the AT END clause of the READ verb and the ON SIZE ERROR option of the COMPUTE, ADD, SUBTRACT, MULTIPLY and DIVIDE verbs will cause a warning message.

5. Internal decimal data items (USAGE COMPUTATIONAL) which are negative will have a sign overpunch indicated over the right most position when it is outputted.

6. Numeric signed items (USAGE DISPLAY), i.e. PICTURE S9(n) are displayed with the appropriate sign overpunch in the lowest order character + or -.

IBLIB (IBJOB Library)

1. Installation optional I/O conversion routines were incorporated in the IBLIB.
Some of the differences with standard conversion routines are described below:

   a. For output, a number converted by E, D, F or I conversion requiring more spaces than are allowed by the field width, will be disregarded and the field will be filled with asterisks (*).

   b. For output with E conversion (D conversion), if the format specification nPEw.d (nPDw.d) requires n + d decimal digits where n + d 8 (n + d 16), n + d - 8 (n + d - 16) zeros will be appended as the low-order digits.
   The information contained in the parentheses pertains to D conversions only.

   c. A normal zero will be written out as 0.0 E-38 or 0.0 D-38 with the number of zeros after the decimal point determined by d in the format statement. In the example given (0.0 E-38), d = 1.

   d. 190 core storage locations are saved as compared with ex-existing routines.

2. Inputting of E, F, D or I type fields has been improved for accuracy.

3. .LXCON (post execution processing routine for COBOL and FORTRAN IV programs) now handles closing of files as follows:

   a. Closed files are ignored unless it is PRINT, PUNCH, or HOLD.

   b. Checkpoint files will be closed with no rewind.

   c. Internal files are not closed.

4. The FORTRAN Mathematics Subroutine Library has been revised to give greater accuracy than previous routines.

5. All double-precision and complex subroutine were reassembled for 7094.

6. The following list of subroutines have been added to the IBJOB library.

COBOL	FORTRAN
.CTAN	FTNC - tangent and cotangent
.CARSN	FASC - arcsine and arccosine
.CGAMA	FGAM - gamma and log-gamma function
.CSINH	FSCH - hyperbolic sine and cosine
.CERF	FERF - error function

7. San Jose Plant site routines added are:

In addition to the plot routines: ANN, AXIS, LINE, NUMBER, BCDFL, BLDFX, PLOTS, PLOT, PLOTE, PLOTEQ, SCALE, SYMBOL, which were in the VERSION 12 Library - we added SGPLOT, SALAX, MAT which will enable the user of the plot routines an easier access to the above. The write-up of these routines, as well as the following ones, can be obtained from Camille Bader, the librarian.
We also added some mathematical routines:
ITRW: For iteration processes, speeds up convergence of equations of the type $x = f(x)$.
RNDMGN: Generate random numbers.
We will, in the future, add more routines to the library, if we feel they will add to our capabilities. As they become incorporated we will circulate memos notifying all users.

GENERAL INFORMATION ON VERSION 13

All subsystems, with the exception of FORTRAN IV, essentially remain the same as in previous versions of IBSYS.

FORTRAN IV users should take note on some differences discovered between VERSION 13 and previous versions.

1. Object decks obtained from previous versions will successfully execute under VERSION 13*
2. Object decks obtained from VERSION 13 will not execute under previous versions.
3. Large programs which previously compiled using other versions may or may not compile using VERSION 13. (In this case, the user may further segment the program or compile using a previous version and executing under VERSION 13).

* Previous version decks using double-precision or complex computations must be compiled using the M94 option.

All IBJOB users should take note that M94 and XR7 are now the assumed parameter of the $IBCBC, $IBFTC, and $IBMAP cards.

VERSION 13 MANUALS

IBSYS Monitor	C28-6248-3
IOCS	C28-6345-3
OPERATOR'S GUIDE	C28-6355-3
UTILITIES	N28-0125
IBJOB Processor	C28-6389-0
FORTRAN IV	C28-6390-0
COBOL	C28-6391-0
MAP	C28-6392-0
IBJOB Processor Debug Package	C28-6393-0
REVISED FORTRAN IV MATH ROUTINES TNL	N28-0154-0

B. Tsuchimoto
M. M. Rogson

BT
MMR/rw

cc: File